VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

# DIPLOMA THESIS

2020 Balakothandaraman Mani

VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

# Text Clustering
# Shlukování dokumentů

2020 Balakothandaraman Mani

VŠB - Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

# Diploma Thesis Assignment

Student: **Balakothandaraman Mani**

Study Programme: N2647 Information and Communication Technology

Study Branch: 2612T025 Computer Science and Technology

Title: Text Clustering
Shlukování dokumentů

The thesis language: English

Description:

The aim of this work is to describe methods for clustering similar text documents. Selected methods will be described, implemented and validated on suitable data sets.

The work will include:
1. Description of document clustering methods.
2. Description of selected algorithms.
3. Design of implementation and implementation of algorithms.
4. Experimental verification of algorithms and their comparison.

References:

[1] Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999
[2] Dan Jurafsky: Speech and Language Processing, Pearson Education, 2014
[3] Bing Liu: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press; 1 edition (June 4, 2015)
[4] Charu C. Aggarwal, Machine learning for text. New York, NY: Springer Science+Business Media, 2018. ISBN 978-3-319-73530-6.

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **doc. Ing. Jan Platoš, Ph.D.**

Date of issue: 01.09.2019
Date of submission: 30.04.2020


doc. Ing. Jan Platoš, Ph.D.
*Head of Department*

prof. Ing. Pavel Brandštetter, CSc.
*Dean*

I hereby declare that this master's thesis was written by myself. I have quoted all the references I have drawn upon.

Ostrava, 15th May 2020 ……..

# Acknowledgement

I would like to thank my supervisor doc. Ing. Jan Platoš, Ph.D, for his advice and support over the period of my thesis and also for his consistent feedback on writing my thesis. Special thanks to RNDr. Eliška Ochodková, Ph.D. for her constant advice and support for my course completion.

# Abstrakt

Tato práce se snaží analyzovat postupy a metody používané pro shlukování textových dokumentů. Také vysvětluje problémy při provádění technik sdružování dokumentů. Seskupování dokumentů budeme provádět analýzou dvou textových datových souborů v reálném světě: 20 diskusních skupin a Reuters, kde 20 zpravodajských skupin bylo rozděleno do dvou variant, přičemž jedna varianta je založena na záhlaví, zápatí a uvozovkách přítomných uvnitř textových dokumentů a druhá varianta obsahuje textové dokumenty bez těchto údajů. Zde budeme hovořit o různých metodách shlukování dokumentů, jejich podobnostech a výzvách při provádění těchto algoritmů shlukování, technikách ověřování kvality shluků a podrobném porovnání. Budeme také diskutovat techniky redukce rozměrů, jejich výhody s jejich podrobným porovnáním. Nakonec diskutujeme a docházíme k závěru, zda tyto metody redukce rozměrů přinášejí v obou těchto algoritmech lepší výsledky.

## klíčová slova

shlukování dokumentů, shlukování textu, 20 diskusních skupin, Reuters, HAC, kmeans, buckshot

# Abstract

This thesis tries to analyse the procedures and the methods used for clustering text documents. Also, explains the challenges in performing the document clustering techniques. We will be performing the document clustering by analysing two real world text datasets: 20 News group and Reuters, where 20 News group has been split into two variants, in which one variant is based on headers, footers and quotes present inside the text documents and the other variant have text documents without these details. Here we will discuss different document clustering methods, their similarities and the challenges in performing these clustering algorithms, its cluster quality validation techniques and its detailed comparison. We will also discuss the dimension reduction techniques, their advantages with their detailed comparison. Finally we discuss and conclude whether these dimension reduction methods produce any better results on both these algorithms.

## Keywords

Document clustering, text clustering, 20 News group, Reuters, HAC, kmeans, buckshot

# Contents

# List of Symbols and Abbreviations

| | | |
|---|---|---|
| BoW | - | Bag of Words |
| DBSCAN | - | Density Based Spatial Clustering of Applications with Noise |
| DENCLUE | - | Density Based Clustering |
| EM | - | Expectation Maximization |
| HAC | - | Hierarchical Agglomerative Clustering |
| IR | - | Information Retrieval |
| IDE | - | Integrated Development Environment |
| LSA | - | Latent Semantic Analysis |
| NLTK | - | Natural Language Toolkit |
| NUMPY | - | Numerical Python |
| PANDAS | - | Python Data Analysis Library |
| PCA | - | Principal Component Analysis |
| RE | - | Regular Expression |
| SVD | - | Singular Value Decomposition |
| TFIDF | - | Term Frequency – Inverse Document Frequency |
| T-SNE | - | t-distributed Stochastic Neighbor Embedding |

# List of tables

# List of figures

# 1 INTRODUCTION

Text mining is rapidly growing in wide range of domains such as healthcare, banking, retail, finance, social media and e-commerce; it is a high demand to understand the usage and the applications of text mining. Text mining is otherwise referred as text analysis, which is the process of converting the unstructured textual data into meaningful and actionable information. Text mining utilizes different machine learning techniques which automatically transform data to generate valuable insights, enabling companies and industries to make data-oriented decisions.

In general, document clustering in text mining [25] is the process of computing large text document collection to find new information to improve specific areas. For large scale businesses like e-commerce, social media and email service providers, the large amount of textual data are being generated every day which is huge challenge and opportunity. On the one hand, these textual data helps the companies and industries to get intelligent insights on product opinions by users or a service which the companies and industries provide. The examples can be potential areas which we use every day in common from filtering spam emails from original emails, social media posts, product reviews and customer feedback from an e-commerce website and support tickets, etc. Text mining plays a huge role on these areas from improvising the quality of service to the overall user's satisfaction.

## 1.1 Motivation

By studying the text document clustering, we gather more information about how similar text documents in a corpus of documents collection are pre-processed and the main differences between a raw text data and a pre-processed data. We should also find how pre-processing works on the unstructured textual data from the text document collection, and also the procedures for pre-processing. We can also find the functionalities of lemmatization, stemming and their differences in clustering and how it can determine any clustering algorithm, then the use of tfidf scores to find the similarities between text documents using some similarity measures.

Another interesting fact is how well an algorithm can detect the clusters between documents by using different similarity measures, whether these similarity measures results better clustering or the combination returns better results from several clustering methods and also to know what all are the other challenges and limitations in performing these algorithms. And the other information to know is by using two variants for 20 news group, whether the use of better tfidf scores based on true group details of the first variant performs well or not compared to the variant 2 which doesn't use the ground truth labels inside their documents (i.e. based on topics).

The next considerable factor is that how a dimension reduction technique can be used effectively, for example how the number of terms can determine the quality of the mentioned clustering algorithms.

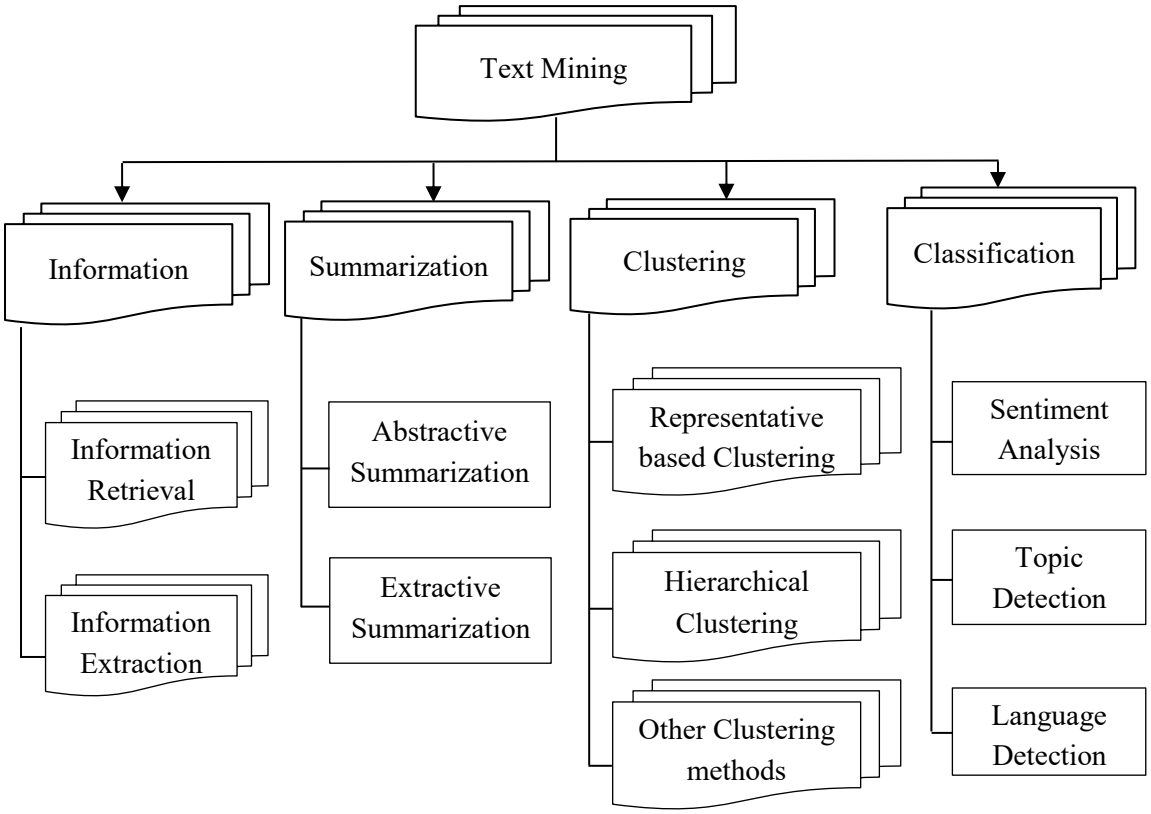


Figure 1: Areas of Text mining

The figure above shows how text mining plays a major role in real world applications and its improvements. In this hierarchy some groups depend on other groups, for example information retrieval depends on some of the document clustering procedures or both methods can be combined as one. [15]

## 1.2 Structure of Thesis

The following chapter begins with Literature review, highlighting the procedures of similar work and research. Chapter 3 describes the detailed theoretical procedures for document pre-processing, clustering methods and its approaches. Following Chapter 4 gives the methodology, architecture and the tools used to support experiments which continue with Chapter 5 experimental phase where

experiments were conducted with various evaluations of arguments, in which followed by the final chapter, Chapter 6 which contains the conclusion from the conducted experiments and ways to extend this work and also future recommendations.

# 2 LITERATURE REVIEW

Document clustering has been in research and development for some time. While Bing Liu [2] discusses more about sentiment analysis and mining options, the Literature [1] from Chris D. Manning and Hinrich Schütze introduced a data representation model using hierarchical clustering method which can cluster frequently used words from the Brown corpus for example. Other literatures like [11] and [16] discusses different text document clustering methods without the usage of any dimension reduction technique to process huge redundant matrixes. The book Machine Learning for Text [4] from Charu C. Aggarwal explained more in detail about the challenges of document clustering methods and also explained two algorithms for clustering similar text documents, in which can be combined to provide better results.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey [15] introduced two new methods fractionation and buckshot which is the combination of Hierarchical clustering and kmeans, both these methods are used in order to browse text using cluster-based approach from a large document collection which comes under information retrieval. Pankaj Jajoo [16] explained more about the challenges in document clustering, other document clustering techniques and quality measures generally used in document clustering. And also proposed a another method called triplet based clustering which uses graph partitioning techniques like by taking similarity scores as edge weights and the nodes are considered as the documents in which if three documents are interrelated to each other, then the triplet is formed also referred as triangular clique in graphs. The advantage of using this triplet clustering method is that the similarity scores gives better clustering results but pitfalls can be what if the number of documents and the number of terms in the corpus increases, then the number of nodes and computational time of similarity score between documents also increases gradually.

The above mentioned works concentrated on working with large text collection to retrieve text using these observations and most frequently used words into clusters, which lacks to cluster document as a whole using dimension methods. But the book [4] from Charu C. Aggarwal explained most of the challenges and limitations for performing document clustering techniques.

In this thesis we will be working on two large datasets named 20 news group and Reuters to cluster their similar text documents and compute clustering quality measures. We will discuss the curse of dimension in large sparse matrixes like tfidf and how it can affect the overall clustering. And here we will compare the most useful way to use dimension reduction techniques and also we will compare two of the document clustering algorithms with different dimension reductions. And finally we will conclude that whether the dimension reduction gives different results on both the algorithms.

# 3 THEORITICAL BACKGROUND

This chapter explains us more to understand what is clustering in general and types of clustering methods and we will discuss what is document clustering and document clustering procedures. We will discuss about the importance of text document clustering and the dimension reduction methods and clustering validity measures used for text documents.

## 3.1 Clustering Methods

Clustering in general means grouping of similar data points. In common there are few types of clustering methods as mentioned below according to the book [8], in this chapter we have used the first two of these clustering methods to compare and evaluate the similar text documents.

- **Representative based Clustering**: This type of clustering technique mainly relies on centroid-based or mediods-based; few examples of this method would be k-means, kernel k-means, EM clustering, etc.

- **Hierarchical Clustering**: There are two main approaches of this method: Agglomerative and Divisive, where Agglomerative method is a bottom-up strategy and Divisive method is top-down strategy. Agglomerative method is widely used and proven strategy. Agglomerative method is otherwise called as HAC, in which it relies on maximum, minimum or average distance between two observations; few examples of this HAC method would be Single Linkage, Complete Linkage and Average Linkage.

- **Graph based Clustering**: In this method, data points are taken as nodes and edges, the weights of the edges are derived from the distance matrix by some distance measures and then the graph cuts and modularity is used to identify clusters; few examples of this method would be Spectral Clustering, Markov Clustering, etc.

- **Density based Clustering**: This type of clustering technique uses nearest density of the data points to identify clusters. The two most commonly used variants of this method are density based and kernel density estimation; few examples of this method would be DBCAN, DBCLUE, etc.

## 3.2 Document Clustering

Document Clustering is a technique that groups' similar text documents such that the text documents in the same group are more similar to each other than the text documents in the other groups. Thus the group of similar text documents are called a Document cluster. The similar text documents are pre-processed well in order to reduce redundant data and then grouped into several clusters based on similarity measures and other criteria that will be discussed further.

## 3.3 Document Clustering Procedures

This section we will discuss about the standard procedures of Text document clustering. Text documents go under feature extraction in other words text documents are generally pre-processed well before execution. For example, a text document may contain more similar words and some similar words can be in the form of upper case, lower case and capitalized. Most of the documents may contain punctuations such as 'Will do! ', 'Are you? '. Hence text data requires a significant amount of pre-processing and so these kinds of things should be removed, altered and normalized before in text document processing, hence this process is also known as Text document Pre-Processing. Further sub-sections we will classify the steps to do document pre-processing procedures.

### 3.3.1 Corpus

First step to perform document pre-processing is the collection of a corpus. A Corpus is otherwise called as "corpora" which is a set of collected text documents or a large set of text document collection like a text database stored online or offline in form of files in a folder, which is often used for statistical analysis and research purposes. Below table shows short example of a corpus with three documents, where $d_1, d_2, d_3$ is considered as documents which is usually in the form of **.txt* files. Real world corpus can contain thousand to million numbers of documents.

| **A small corpus with three documents** |
|---|
| $d_1$ = "This is the 1st document" |
| $d_2$ = " document 2 starts from here" |
| $d_3$ = "third document considered as example" |

Table 1: Example of a Corpus

### 3.3.2 Tokenization

Tokenization is an important process in text document pre-processing and clustering. Tokenization is the parsing process of a raw text document into several tokens (i.e. words). As per convenience, tokenization can be performed as first step or at the last after case folding, removing stop words and punctuations. For example, there is a text document containing few words, each word is assigned as its separate tokens. Consider we have three documents named $d_1$, $d_2$, $d_3$ these document tokens can be represented as below:

| Document(s) | Tokens |
|---|---|
| $d_1$ = "This is the 1st document" | "This" "is" "the" "1st" "document" |
| $d_2$ = "document 2 starts from here" | "document" "2" "starts" "from" "here" |
| $d_3$ = "third document considered as example" | "third" "document" "considered" "as" "example' |

Table 2: Example of Tokenization

### 3.3.3 Case Folding, Stop words and punctuations

The process which changes the upper case words into lower case words called case folding. In most cases lowering the words can be taken as complementary because changing upper case into lower case has no effect. But in some cases it is advised not to change the lower case words into upper case because, Upper case words can also be used to extract titles from a particular document. But in this paper we have lowered the upper case words since our aim is to find only clusters between documents.

Stop words are the set of words that appear very frequently but don't have a lot of informational content or sometime meaning. Filtering stop words doesn't cut out any information density or the meaning of our main text document. Then the remaining set of words goes to stemming process. Below table 3 shows few stop words commonly used in most documents while removing these words not only reduce the dimensionality but it also doesn't change the meaning of any sentence or document context. Along with stop words, single characters words and extra white spaces has to be removed.

| | | | | |
|---|---|---|---|---|
| I | am | myself | is | are |
| me | be | did | was | were |
| not | been | but | his | she |
| for | it | because | about | with |
| you | him | her | whom | what |

Table 3: Few Stop words

### 3.3.4 Handling hyphenated words

A hyphen (-) is a punctuation mark used as a split between set of words. Hyphens in a sentence of a document can be very tedious to process. In some cases separating a hyphen in a word can give different results to the word's meaning compared to the main hyphenated word. For illustration consider an example with a compound word: vice-president, changing this single word into two words such as "vice" and "president" might not be good idea for text pre-processing, because splitting into two words yields different results in document term. For example, consider a sentence: "This tiger is nine-year-old"

In the above case the hyphenated word "nine-years-old" defines a single meaning in which tiger is nine years old. Splitting into three words might be needless. Few examples of hyphenated words which can change the complete context or the meaning of an entire sentence are: "Brother-in-law", "son-in-law", "high-tech", "bottom-up", "build-up", "top-down", "high-speed" and "room-mate"

Some words might have these three common prefixes: pre, mid and post. Such words are: "pre-ponded", "post-ponded", "mid-range". Because of this reason we keep the original hyphenated words, for example 'bottom-up' will be changed into 'bottomup' to maintain consistency of the words.

### 3.3.5 Stemming Process

To reduce the complexity and improve quality of term frequency data stemming process takes place in one of the pre-processing steps. For example in the table below we have different words with same suffixes, but it yields different stems based on letters next to the word's suffixes. In other words stemming is the process to reduce terms to their stems in Information Retrieval. Stemming is a crude process of chopping the suffixes in a particular word. For example: if we have three words 'eat', 'eating' and 'eaten' appearing in a document collection. All these three words yields the same meaning which is 'eat', hence by performing stemming, all three words reduced to 'eat' which therefore increases the frequency of that particular word statistically and which also decreases the initial dimensions of tf-idf that will be discussed further.

| Form | Suffix | Stem |
|---|---|---|
| magically | -ally | magic |
| activities | -ies | activ |
| magnitude | -e | magnitud |
| equal | -al | equal |
| usual | -al | usual |
| unusual | -al | unusu |
| tensely | -ely | tens |
| understanding | -ing | understand |
| carrying | -ing | carri |
| fraudulentness | -ness | fraudul |
| carelessness | -ness | careless |
| pennies | -es | penni |

Table 4: Example of Stemming Process

### 3.3.6 Lemmatization

Lemmatization is sometimes called as another form of stemming process. Even though stemming cops off the word to reduce the word's length, Lemmatization is like a dictionary which does not change the meaning of any word. For example: the word 'better' is converted into 'good' in lemmatization process. And the word 'driving' will be cut into 'driv' in stemming but in lemmatization it is converted into a word 'drive'. Lemmatization applied to a word returns proper root word in which the applied word belongs to a proper language. In Lemmatization root word is called **Lemma**. A lemma is a canonical form (plural lemmas or lemmata), citation form or dictionary form of a set of words. Another example for lemmatization is for the word: '*Disconnect'.* Consider a dictionary contains a 'key' called '*Disconnect*' and its four 'values'.

{'*Disconnect'*: ['Disconnect', 'Disconnections', 'Disconnected', 'Disconnecting', 'Disconnection']}

Hence the word '*Disconnect'* is the root word for all other four words i.e.: *Disconnections, Disconnected, Disconnecting, and Disconnection*.

### 3.3.7 Stemming vs Lemmatization

As we can see from the above examples stemming is little straight forward but lemmatization is like a dictionary finding the main root word based on the input value. Hence stemming process is a bit fast compared to lemmatization. Below table shows the differences between stemming and lemmatization process, we can conclude that stemming process just chops down the suffixes of each word, but lemmatization gives the exact meaning even if the word is in plural form or even the word is in present tense.

| Word | Stemming | Lemmatization |
|---|---|---|
| was | wa | be |
| eating | eat | eat |
| stated | state | state |
| starving | starv | starve |
| connections | connect | connect |
| nothing | noth | nothing |
| volunteer | volunt | volunteer |
| celebration | celebr | celebration |
| honorary | honorari | honorary |
| caring | car | care |

Table 5: Stemming vs Lemmatization

In the above example the word '*Caring*' in the stemming process becomes '*car*', if there are text documents containing the word '*car*' which refers to an automobile, this will lead to redundant

number of unique words and also documents cluster which are not relevant at all. Since lemmatization reverts back to the original root word '*Care*', lemmatization preserves the original meaning, hence we can further reduce the number of meaningful unique words in the entire corpus for post-processing, in this paper we have used WordNet Lemmatizer, which is explained in the chapter 4.2.

### 3.3.8 Bag of Words (BoW)

After performing the above pre-processing steps, set of text (i.e. words) can be considered as one form called Bag of words, which is nothing but putting each and every unique words from all the documents into one complete list of words. In natural language processing, bag of words is considered as the process of converting each document into a vector. Hence it is also called as Vectorizer. Bag of words is considered as the column names for TF-IDF matrix formation. Below table 6 shows the example of how *n* number of documents can be converted into one term called "bag of words'':

| **Bag of Words for table 2** |
|---|
| "1st" "document" "2" "start" "here" "third" "consider" "example" |

Table 6: Example for Bag of Words

### 3.3.9 TF-IDF Calculation

Text documents are raw text data which will not be in machine understandable format; hence we need to convert the text documents into a numerical format, for this we need to use the Term Frequency (TF) and Inverse Document Frequency (IDF). TF-IDF is the fundamental building block for many search algorithms. This term is mostly used in Information Retrieval Systems and Text Mining [22].

***Term Frequency***: Term Frequency is the measure of how frequent a word (i.e. term) occurs in a document. Each and every text document is different from one another based on the length and context. So frequency of a term will be slightly higher if the document length is higher.

- A word that occurs most frequently is most probably the important word in a document.

$$\text{TF (t)} = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}} \tag{3.1}$$

***Inverse Document Frequency***: Inverse Document Frequency is the measure of how often a word occurs in an entire set of documents.

- This shows how well common words appear everywhere in the entire set of documents

$$\text{IDF}(t) = \log_{10}\left(\frac{\text{Total Number of documents}}{\text{Number of documents with term t in it}}\right) \quad (3.2)$$

$$\text{Then, } \textit{TF-IDF} = \text{TF}(t) * \text{IDF}(t) \quad (3.3)$$

Higher the TF-IDF score lowers the importance of a particular word. Lower the TF-IDF score highers the importance of the word. TF-IDF is the starting point of any text analytics. This data is mainly used to convert raw text of documents into numerical data for further calculations which is used for some machine learning algorithms.

### 3.3.10 Applications of TF-IDF

TF-IDF is widely used in the field of Text mining. And TF-IDF score is the statistical measure used to determine how the word closely connected to a document in a collection of text documents (i.e. corpus). This scoring scheme is often taken by the many systems as a central tool for

- **Information Retrieval Systems**: In general, these systems have large data collected and stored in the form of indexes, ranking model and some machine learning algorithm. Hence if a user searches for a query it retrieves the exact related information. Few IR systems can be: Web Search, Desktop Search and etc.

- **Topic Modelling or Topic Extraction**: This is the process of discovering the topic which is previously unknown in a set of collection of texts, without a prior knowledge. There are two types of approach followed in topic modelling: Conceptual approach and Generative approach.

- **Automatic Text Summarization**: This is the process of shortening a large text document with a program or software. Creating a summary from the whole document only picks the major points from the original document. The most important feature in this process is it tells the most important things in a short amount of time. There are two types of summarization in general, Extractive Summarization and Abstractive Summarization.

- **Sentiment Analysis**: The process of computationally identifying and categorizing opinions of a user from few text and whether also the particular topic is positive, negative or neutral. Sentiment Analysis comes under Text Classification. Example would be customer ratings in e-commerce websites.

- **Recommender Systems**: These systems are used in e-commerce and online streaming applications. Recommending users with the most relevant and related contents. These systems use two basic filtering techniques to recommend content which are: Content based filtering and Collaborative Filtering. Another type of recommender system is Hybrid Recommender System. This basically recommends the content based on other user's preferences

# 3.4 Document Similarity Measures

In this chapter we will discuss the different similarity measures [8] used for clustering text documents and its comparison with each other.

## 3.4.1 Euclidean distance

Euclidean Distance is the distance between data points. It is also called as L2-norm Euclidean Distance and it is calculated from the equation below:

$$Euclidean\ Distance = \delta\ (x, y) = \|x - y\| = \sqrt{(x-y)^T(x-y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

## 3.4.2 Cosine Similarity

In general, Cosine similarity is the smallest angle between vectors. Consider X and Y are the two vectors; the similarity can be calculated using formula below:

$$Cosine\ Simlarity = \cos(\theta) = \frac{X.Y}{\|X\|.\|Y\|} = \frac{\sum_{i=1}^{n} X_i\, Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\ \sqrt{\sum_{i=1}^{n} Y_i^2}}$$

Thus, the cosine of the angle between X and Y is given as the dot product of the unit vectors X and Y. Here consider we have documents X…*to* N, the cosine similarity can be obtained by using TF-IDF matrix, which will be dimension reduced and cosine similarity is computed between documents as one of the distance measure which will be discussed in Buckshot 3.6.3.

### 3.4.3 Jaccard Similarity

Jaccard similarity is sometimes called as Jaccard coefficient (or) Intersection over Union is the measure of two sets to see which is commonly shared and which are distinct. Consider two documents named X and Y, then their Jaccard similarity is calculated as:

$$Jaccard\ Similarity = J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

### 3.4.4 Cosine Similarity v/s Jaccard Similarity

Cosine similarity takes total length of the words, these words are basically comes in the form of TF-IDF matrix but Jaccard similarity takes only the unique set of words from each document. This means if we have any number of repeated words in a document, cosine similarity changes but Jaccard similarity does not change at all. Hence Jaccard similarity is useful for the cases where the repeated words are not much important in the documents, while Cosine similarity is useful where the repeated words are important in the documents [20].

For the smaller text document collections it is always better to use Jaccard similarity because the repetition of any word doesn't reduce the document similarity. And for larger text document collections where duplication of words matters, cosine similarity can be chosen.

Let's consider the same example we used for Jaccard Similarity Calculation.

$d_A$ = "there was a blue tulip in the garden yesterday"
$d_B$ = "one little Cloud out of the blue sky"

In order to calculate Jaccard similarity, consider that we reduced the total number of words by removing stop words from both the documents A and B. The Jaccard score is calculated by the formula in 3.4.3, further these documents can be represented in the form Venn diagram. Thus the similarity between these two documents would be 1/ (4+6-1) = 1/9 = 0.111
Where |A| = 4, |B| = 6, |A ∩ B| = 1.

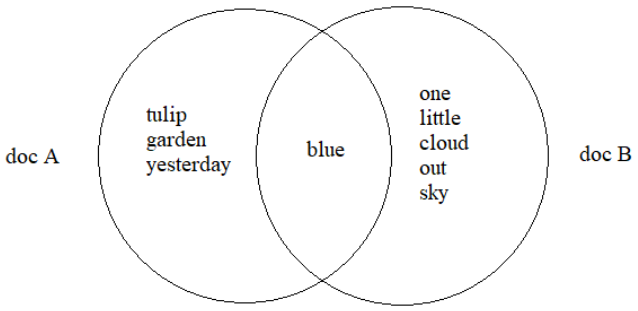


Figure 2: Venn diagram for Jaccard Similarity

So we can notice that the Jaccard similarity can be calculated straight from the documents but for the cosine similarity we need the help of TF-IDF matrix. Here we have used cosine similarity as one of the distance measure since the duplication of words gives better tfidf in which leads to better clustering results.

## 3.5 Dimensionality Reduction Techniques

This section deals with the procedure and methods used for matrix factorization and dimension reduction. Dimensions are always been the problem while performing clustering techniques. The text documents collected from the corpus usually converted into a large tfidf matrix m x n (rows and columns), this tfidf matrixes usually contains lot of zeros which are redundant for post- processing. Also the corpus might have huge number of dimension (documents x number of terms) for processing it is always tedious to process the tfidf matrix. Also these matrixes are highly correlated with each other. So in order to lower its correlation and also to represent data it should be converted into lower-dimensions. This process is often termed as the dimensionality reduction. Each and every dimensionality reductions can be expressed as low rank factorizations of its own document term matrix or TF-IDF matrix.

### 3.5.1 Truncated-SVD

In general, the regular SVD produces a factorization technique that factors a matrix X into three different matrixes U, V and Σ. SVD technique is very similar to PCA method, but the difference is SVD method factors the data matrix, but the PCA method factors the covariance matrix. And truncated-SVD factors where the number of columns is specified beforehand. For example truncated-SVD will produce matrices with specified number of columns in our case number of terms, whereas in the SVD given an n x n matrix which will produce matrix with n columns. One of the main advantages of using truncated-SVD is that it can work with sparse matrixes while t-SNE and PCA cannot operate, because the covariance matrix must be computed for PCA, which requires operating on the entire matrix.

### 3.5.2 T-SNE

T-SNE is a t-distributed Stochastic Neighbor Embedding is an unsupervised non-linear dimension reduction technique primarily used for exploring the data and visualizing the high-dimensional data [23]. The t-SNE dimension reduction technique usually calculates the similarity measure between pairs of instances in the low dimensional data and in the high dimensional data. The difference between t-SNE and PCA is t-SNE preserves only the small pairwise distances or local similarities whereas the PCA preserves the large pairwise distances to maximize variance [24]. Hence t-SNE provides a best practice for visualizing high dimensional data. Disadvantages of using t-SNE are: t-

SNE requires dense matrix hence cannot work with sparse matrixes, the time and space complexity suffers heavily since t-SNE has a quadratic time and also uses resources heavily and can be only used for plotting 2d plots.

## 3.6 Document Clustering Methods

This section will discuss the methods used for text document clustering. In general, Clustering is an unsupervised learning technique. Clustering methods can be either flat or hierarchical. Text document clustering is the process of partitioning a corpus into several groups based on similar text between documents. The two methods used here are k-means clustering and agglomerative hierarchical clustering which will be discussed further.

### 3.6.1 K-Means Clustering

One of the famous clustering techniques is K-means clustering. The objective of k-means clustering in text documents is to process the partitioning of similar text documents into several clusters. In general *k*-means clustering takes k as the input value, where *k* is the number of clusters to return after clustering. Another input parameter can be given for the number of iterations to refine the centroids to obtain results.

By utilizing matrix factorization technique like singular value decomposition (SVD) we hence reduce the large dimensions of sparse TF-IDF matrix into smaller terms, here terms is defined as the number of words or number of columns. SVD in text document approach is always referred as truncated-SVD which was discussed in detail from the section 3.5.1. The remaining matrix with n terms has been used to process the k-means clustering.

#### 3.6.1.1 K-Means Algorithm

Step 1: Specify the value of k, i.e. k is the number of clusters.
Step 2: Randomly select the initial centroids.
Step 3: Assign the objects which are closer to the centroid.
Step 4: Recalculate the centroid with the k value.
Step 5: Repeat steps 3, 4 and stop if there is no change in centroids.

The k-Means algorithm usually runs on the time complexity of O ($n^2$), where n is considered as the number of documents present in the corpus. And the input value k is the number of clusters needed. The main advantage of kmeans clustering algorithm is that it works well on larger datasets and small pitfall for kmeans is considered to be bad seed selection.

### 3.6.1.2 Optimal Number of Clusters

In general, finding the optimal number of clusters is an effective way used for k-means clustering algorithm for partitioning with perfect *k*-value. Commonly used methods to find the optimal number of clusters are elbow method and silhouette score. Elbow method is the calculation of sum of squared distances from each data point (i.e. documents) assigned to its respective cluster. Elbow method shows the lower score at the point where there is not enough deviation but silhouette method is a contrast of elbow method in which it shows the highest score where optimal number for *k* can be found. Elbow method can be used if k value is not determined before clustering; but in our case we know the true k value for the datasets we used here for the experiments.

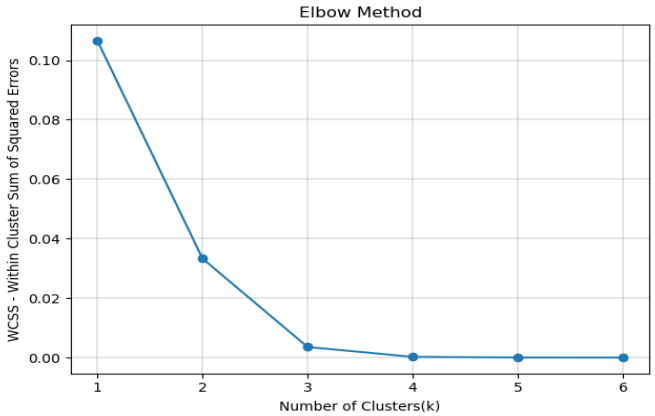


Figure 3: Example of using Elbow method

## 3.6.2 Hierarchical Clustering Methods

Hierarchical Clustering is another clustering technique, which groups similar data points in our case grouping similar text documents. Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy. There are two types of hierarchical clustering methods; one is agglomerative clustering and the second is divisive method.

In Agglomerative clustering or bottom up clustering method we initially assign each data point as an individual cluster. At each iteration the similar clusters are merged with other clusters based on its similarity until we left with one cluster. On the other hand, Divisive method or top down clustering method is complete opposite of agglomerative clustering method, where we assign all the observations to a single cluster and then group the cluster to two least similar clusters. And the process is carried out recursively on each cluster until there is one cluster for each observation.

Here the TF-IDF formation (i.e. matrix) can be used to calculate the cosine similarity between similar text documents. This will produce a new matrix called cosine similarity matrix where its diagonals will be 1 which means document '*x*' is similar to its own document '*x*' itself. Using the cosine similarity matrix the agglomerative hierarchical clustering technique works really well for

smaller datasets, but for larger datasets it is not recommended which will be discussed further in 3.6.2.5.

### 3.6.2.1 HAC Algorithm

Step 1: Assign each document to its own cluster.
Step 2: Calculate the distance between each set of clusters.
Step 3: Merge two objects based on the minimum or maximum distance between them.
Step 4: Calculate the distance between new cluster and each of remaining clusters.
Step 5: Repeat the steps 2 to 4 until all the object becomes one whole cluster.

Here object represents documents and merging two documents refers merging two document based on their similarity scores. The similarity matrix is the first parameter initiated in this method and then we have to choose the number of clusters before starting the iteration, hence the iteration will be stopped when we achieve the given n number of clusters. There are few ways to merge two documents into a cluster by following techniques.

### 3.6.2.2 Single Linkage Clustering

In general, single linkage calculates the shortest distance between any object of one cluster to any object of the other cluster. In our case the maximum cosine score is the shortest distance between two similar text documents. The maximum the cosine score means the maximum possibility that the two documents are similar to each other. As suggested in the book [4], in single linkage clustering, the similarity between the two nearest pair of documents from $C_i$ and $C_j$ is used to be the similarity between $C_i$ and $C_j$. Therefore, if $s_{ij}$ is the similarity between clusters $C_i$ and $C_j$, then we get

$$s_{ij} = MAX_{\bar{X}\in C_i, \bar{Y}\in C_j}\, cosine\ (\bar{X}, \bar{Y})$$

### 3.6.2.3 Complete Linkage Clustering

In this method, the distance between two objects is considered as the longest distance between any object of one cluster to any object of the other cluster. In our case the minimum cosine score is the maximum distance between two similar text documents, hence these two documents are dissimilar to each other. Present in the book [4], in complete linkage clustering, the similarity between the two most dissimilar pair of objects from $C_i$ and $C_j$ is used. Therefore, if $s_{ij}$ is the similarity between clusters $C_i$ and $C_j$, then we get

$$s_{ij} = MIN_{\bar{X}\in C_i, \bar{Y}\in C_j}\, cosine\ (\bar{X}, \bar{Y})$$

Both single and complete linkages have their pitfalls, which is both these methods depend only one single pair of documents to determine the similarity. Hence group-average linkage is considered to perform better which will be discussed further.

#### 3.6.2.4 Group-Average Linkage Clustering

In the group-average linkage clustering, the average similarity between all the documents in the cluster $C_i$ and in the cluster $C_j$ is calculated and then clusters $C_i$ and $C_j$ have their similarity $s_{ij}$

$$s_{ij} = MEAN_{\bar{X}\in C_i, \bar{Y}\in C_j}\ cosine\ (\bar{X}, \bar{Y})$$

#### 3.6.2.5 Pitfalls of Hierarchical Clustering

The main disadvantage of hierarchical clustering algorithms is its expensiveness and space complexity which is discussed in detail from the book [4]. For example: consider a corpus containing *n* documents, then the space complexity to compute similarity matrix by this method would be $O(n^2)$. The initial input parameter for this algorithm is the similarity matrix and the merging two similar documents make O(n). The space complexity is often considered as a pitfall in this method, if the corpus contains million numbers of documents then it would take terabytes of space from the memory. Also the re-computation of similarity between two documents for merging makes the time complexity $O(n^2+n)$ which make even worse $O(n^3)$. One main advantage of using hierarchical is it works well for smaller datasets. So hence we have used the combination of hierarchical and k-means to verify whether we obtain better clustering.

### 3.6.3 Combination of Hierarchical and K-Means

Both the methods have advantages and disadvantages in terms of execution time and accuracy. The k-means algorithm works well on the larger datasets while it suffers if the seed is poor. The Hierarchical clustering algorithm on the other hand is expensive and works well for smaller datasets. This suggests we can merge hierarchical clustering with k-means in order to get better results of both these algorithms.

#### 3.6.3.1 Buckshot

The Buckshot is a two phased hybrid approach where it uses both hierarchical agglomerative clustering and k-means clustering. As suggested in the books [4] and [15], this observation suggests that a hierarchical method can be merged from a comparatively small sample of documents to a strong set of k-clusters, whose centroids are often will be excellent to create a superb seed set for the k-means algorithm. This results of two-phase approach in which the primary phase uses hierarchical clustering

method and then the second phase uses k-means clustering algorithm. The goal is to reduce the processing time by using HAC & kmeans on larger datasets. Since random sample subset is used, this method is not deterministic, thus repeated execution produces different results, but mainly it avoids bad centroid selection.

#### 3.6.3.2 Buckshot Algorithm

Step 1: Randomly select *d* documents; where *d* is the sample subset (i.e. $\sqrt{n}$ or$\sqrt{kn}$); *k* is the number of clusters and *n* is the total number of documents present in the corpus.
Step 2: Calculate *k* clusters in this subset *d* using hierarchical agglomerative with group average clustering.
Step 3: Determine the centroid of each of the *k* clusters obtained from the subset *d*.
Step 4: Using the above centroid, apply kmeans for the entire document corpus.
Step 5: Obtain k clusters.

In this paper we have used $\sqrt{kn}$ for the selecting the subset of documents *d* and hierarchical clustering with group average linkage, since Buckshot with group average linkage works better compared to Hierarchical clustering with single or complete Linkage.

## 3.7 Clustering Validation

Generally the clustering algorithms are evaluated by two types of validity measures: Internal validity measures and external validity measures. The internal validity measure doesn't require true labels of the groups. The reason we choose external cluster validity measures over internal validity measures is that, internal validity requires a criteria derived from the objective functions of various clustering algorithms. Hence it becomes impossible to compare the effectiveness of any two different clustering algorithms. Examples of internal validity measures are Silhouette Coefficient, Calinski-Harabasz index, Davies-Bouldin index etc.

### 3.7.1 External Measures

The External validity measure of clustering requires the true labels to evaluate the clustering quality. Even though the true labels are required before clustering, it won't be used in any of the clustering algorithms. Therefore the measuring name is called external, since it is inherently external to both the clustering algorithm and the data. And these clustering algorithms usually generate new labels to the objects, the original labels and the obtained labels are compared for several external clustering measures which will be discussed further.

### 3.7.1.1 Homogeneity

In homogeneity external validity measure, the clustering algorithm must assign only the labels of a single group. This means the group distribution of each and every cluster should contain only the group labels of true labels, which makes entropy equal to 0. To determine the homogeneity of a clustering algorithm, first conditional entropy of the group distribution is computed. Homogeneity score should be between 0 to 1, where 0 considered as bad score, 1 is considered as perfect score and scores close to 1 is considered as better clustering results. Non perfect labelings can also be perfectly homogeneous, for example: the true labels of a sample contains only two groups (i.e.: 0 and 1) and contains 4 observations [0, 0, 1, 1] and if the algorithms predicted 3 clusters [0, 1, 2, 3], and then the homogeneity score will be still 1. And if the clusters have samples from different clusters are not considered as prefect score, for example: true labels [0, 0, 1, 1] and predicted labels [0, 1, 0, 1] would score 0.

$$H = 1 - \frac{H\,(C \mid K)}{H\,(C)}$$

Where *H(C | K)* is the conditional entropy of the groups predicted and computed as:

$$H\,(C \mid K\,) = -\sum_{C=1}^{|C|}\sum_{k=1}^{|k|}\frac{N_{\,C,k}}{N} * log_{\,2}\left(\frac{N_{C,k}}{N_k}\right)$$

Here N is the total number of documents; $N_k$ is the cluster k and $N_{C,\,k}$ number of documents from group C assigned to cluster K, further we will discuss the computation of entropy (i.e. H(C) or E).

### 3.7.1.2 Completeness

Completeness is considered as all documents of a given group are assigned to the same cluster. Completeness score also lies between 0 and 1.

$$C = 1 - \frac{H\,(\,K\,|C\,)}{H\,(K)}$$

### 3.7.1.3 V-measure

V-measure is another entropy based measure which externally computes how well the homogeneity and completeness have met their criteria. Hence, V-measure is the harmonic mean of homogeneity and completeness. V-measure also ranges from 0 to 1 and it is computed by:

$$V = 2 * \frac{H * C}{H + C}$$

### 3.7.1.4 Entropy

Entropy is mostly used to evaluate the disorderness of the cluster objects [19]. While predicting the group labels, entropy measures the uncertainty of the predicted labels by the clustering algorithm. Let us assume estimating the entropy *E (j)* of a specific cluster *j*, then the entropy of cluster *j* can be calculated using the below equation:

$$E_j = -\sum_{j=1} p_{ij} log_{2(}p_{ij)}$$

The sum of all entropies of every cluster weighted by the size of each cluster is the total entropy and it is calculated below equation. The total entropy is often referred as conditional entropy and the lower value of conditional entropy indicates higher quality clustering. And conditional entropy score always lies in between (0, $\log_2 (k_d)$)

$$Entropy\ (E) = \sum_{j=1}^{k_d} \frac{n_j * E_j}{n}$$

Where $k_d$ is the number of total clusters, $n_j$ is the $j^{th}$ cluster size and n is the total number of documents.

### 3.7.1.5 Purity

Purity is an intuitive way to measure the cluster quality. But it heavily relies on most used label on a particular cluster and it relatively ignores the distribution of the other remaining labels. A way to estimate the clustering purity is by the use of contingency matrix, where the rows $c_i$ would be number of clusters and columns $t_j$ would be number of classified objects in our case text documents. Then for each cluster $c_i$, maximum value of its row is taken and added together; finally the total is divided by the total number of documents (N). The purity score usually in the range from 0 to 1 and the score which is close to 1 is considered as better quality clustering.

$$\text{Purity (P)} = \frac{1}{N} \sum_{i=1}^{k} max_j \left| c_i \cap t_j \right|$$

Where *N* is the total number of documents, *k* is the total number of clusters, $c_i$ is the cluster present inside *C* and $t_j$ is the number of classified text documents.

# 4 METHODOLOGY

This chapter we will discuss more about the methodology mainly used for text data pre-processing, clustering and visualization. Also we will discuss about the working process of the experiments in detail on the experiments section 4.2.9.

## 4.1 Architecture

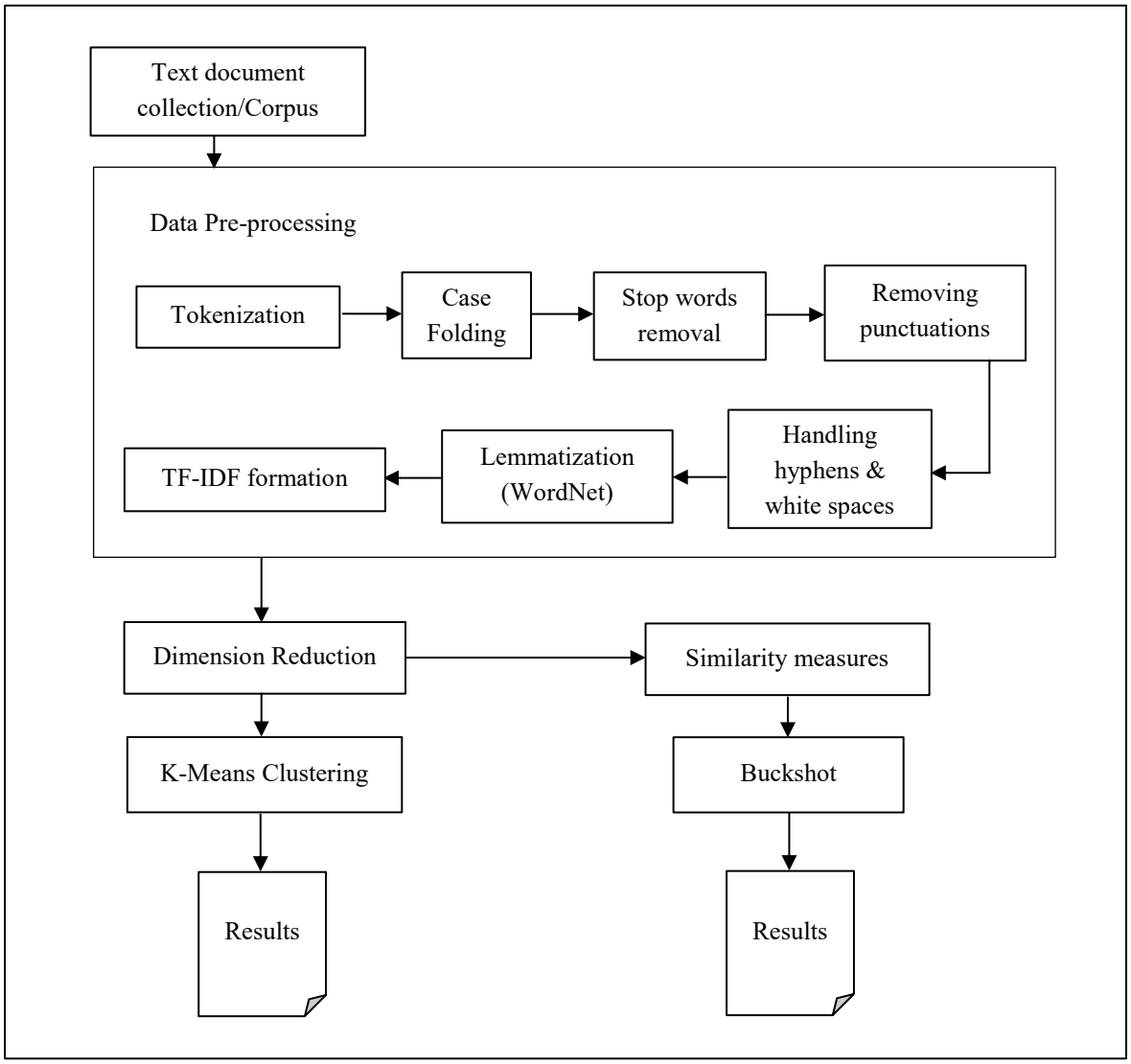


Figure 4: Document Clustering Flow Diagram

## 4.2 Tools and Packages

For experimental purposes we have included few packages, which are listed in the order from data wrangling, pre-processing, post-processing and finally for the visualizations.

### 4.2.1 Anaconda: Jupyter Notebook: IDE

Anaconda is an open source tool to work with Python and also manages libraries efficiently. It is compatible with most operating systems like Linux, Windows and macOS. Anaconda comes with predefined libraries, which is easy for managing multiple development environments also libraries. It also contains other tools like Visual Studio code, Glueviz. Syder, etc. [18]

Jupyter Notebook is an IDE used to integrate seamlessly with python. The best feature of Jupyter notebook is its interactive development environment. Even though some complicated calculations require some processing time such as displaying results and plotting images, Jupyter Notebook solves this and returns the results and images instantly. Another effective feature present in Jupyter Notebook is saving the previously executed results and plots in an html file format which can be later used for future reference.

### 4.2.2 OS: File System management

OS module provides all the operating system functionalities like creating, renaming, deleting, moving files and folders. It acts as an intermediate between operating system and python, where one can list all the files in a directory or a folder. Here we have used OS module for the Reuters dataset for data wrangling and for the creation of pandas dataframe from reading files and sub folders from the main root folder.

### 4.2.3 Pandas: Data Management

Pandas is a widely used tool in python, where we perform data wrangling. Data wrangling is the process of cleaning up data before pre-processing. The data management and modification are the two most useful features in pandas. We used pandas to import all text files into one dataframe and perform some statistical analysis. Here we used pandas dataframe which handles huge number of rows, rows are generally text files, which further processed into pre-processing functions which later converted into a tfidf matrix for further processing.

### 4.2.4 RE: String Management

RE is a regular expression library used for cleaning text data before processing. Regular expressions (also called as called REs, or regexes, or regex patterns) are essentially a small, highly specialized programming language embedded inside Python and made available through the re library. Using this tiny language, one can simply specify to match the principles for the set of possible strings. We have used regular expression library for text with the patterns like extra white spaces at the beginning or at the end of a sentence and also removing stop words and punctuations.

### 4.2.5 NLTK: Lemmatization

For lemmatization we have used WordNet lemmatization from NLTK. WordNet lemmatization is a large publicly available database which helps to establish connection between set of words using structures semantics. It uses most commonly used lemmas for a given set of words. The WordNet database can also be downloaded or updated using nltk.download('wordnet') function.

### 4.2.6 Sklearn: Dataset and Dimension Reduction

Sklearn is a scientific library, which provides most famous datasets and dimension reduction functions. We have used 20 news group for the dataset, tfidf vectorizer for generating tfidf sparse matrix and truncated-svd for the dimension reduction. Sklearn also provides pairwise distance measures and cluster validity measures like homogeneity, v-measures and completeness [21].

### 4.2.7 Numpy: Array management

Numpy is a scientific computing core library in python. It provides support for a high-performance array management and tools for managing large arrays. Since Numpy library was written in C, the compiled functions and functionalities run as fast as possible, so it can be used for many big data applications. Also it is a proven array library for improving time complexity of array computation over lists (or) lists of lists in python. Numpy library can also use to create random arrays, arrays with zeros and arrays with ones for the purpose of testing. The main advantage of using numpy is: numpy usually saves array in a binary format which is readily accessed by the operating system which makes numpy convenient, fast and less memory occupancy.

### 4.2.8 Matplotlib: Visualization

It is always difficult to visualize textual data especially with some text and numbers after the clustering is done. In order to visualize we have to convert them into data points and plot it using some visualization library. For this we have used Matplotlib for plotting graphs.

### 4.2.9 Miscellaneous

We have also used few python functions like translation. Translate is a function used to translate a string if there is a match in the expected pattern. Here translate method used for removing all new lines: "\n" and tabulated spaces: "\t" in the corpus data for pre-processing. We also used a math function for logarithm base 2 operations (i.e. $\log_2(x)$) for entropy calculation and deepcopy for assigning similarity matrix into new variable in memory to compare distances with main similarity matrix for HAC group-average linkage algorithm.

# 5 EXPERIMENTS

This chapter discusses the experiments conducted with the selected dataset. The following sections shows the properties of the dataset and various combinations used to redefine the cluster results. Each evaluation of results with these combinations is represented in tables and figures with various criteria.

## 5.1 Experimental Process

The experimental process has been considered with six different processes as mentioned in the below diagram. The first phase of the process is text data collection, next process is data wrangling which makes raw data into most useful format, then the text document pre-processing with procedures mentioned in 3.3., Then the clustering techniques followed to measure clusters are mentioned in the previous chapters. Finally the experiment conclusion is evaluated with the combination of various obtained results.

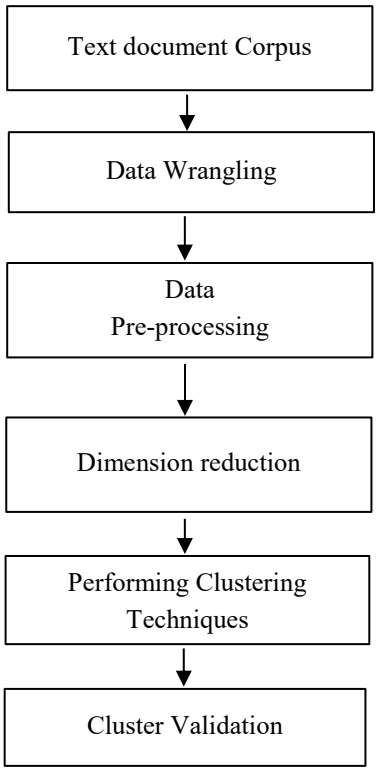


Figure 5: Work flow of the experiments

The sparse tfidf matrix usually contains most number of zeros which is redundant to the any clustering process; hence it is better to reduce the number of terms using truncated-SVD since it handles sparse matrixes well. After pre-processing the data we have sparse tf-idf matrix, which further goes into dimension reduction and clustering methods as mentioned below.

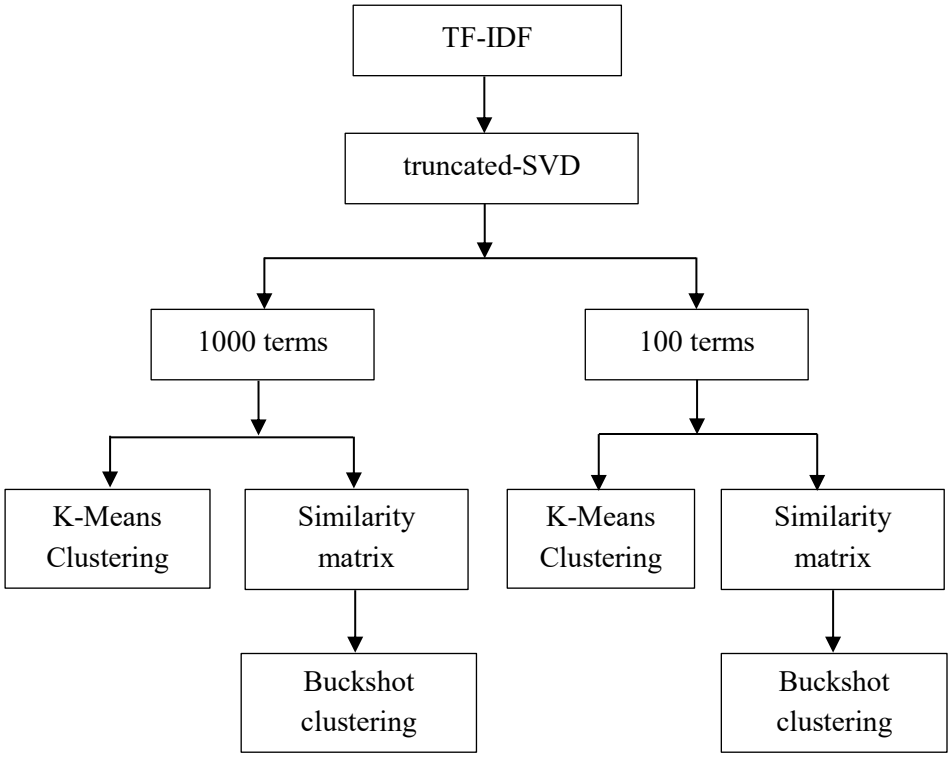


Figure 6: Methodology after TF-IDF

Here similarity matrix uses cosine distance measure which is the preliminary input for buckshot clustering

## 5.2 Datasets

As a text document corpus, we have selected two datasets 20 Newsgroup and Reuters version 1.0 [17]. These two datasets are the most popular datasets for experiments in text document clustering, classification and machine learning applications. The following sections show the dataset properties and their distribution of files.

### 5.2.1 20 News Group

In 20 news group there are 2 variants, one with headers, footers and quotes and the other without headers, footers and quotes which can be discussed further. The table 5.2 below shows all the descriptive properties of the dataset.

| Category | Details |
|---|---|
| Name | 20 News Group |
| Original File size | 34.9 Mb |
| Data type | Text |
| Data source | qwone |
| Total number of files | 18846 |
| Avg. file count in a folder | 942.3 |
| No. of folders based on groups | 20 |

Table 7: Properties of 20-News Group dataset

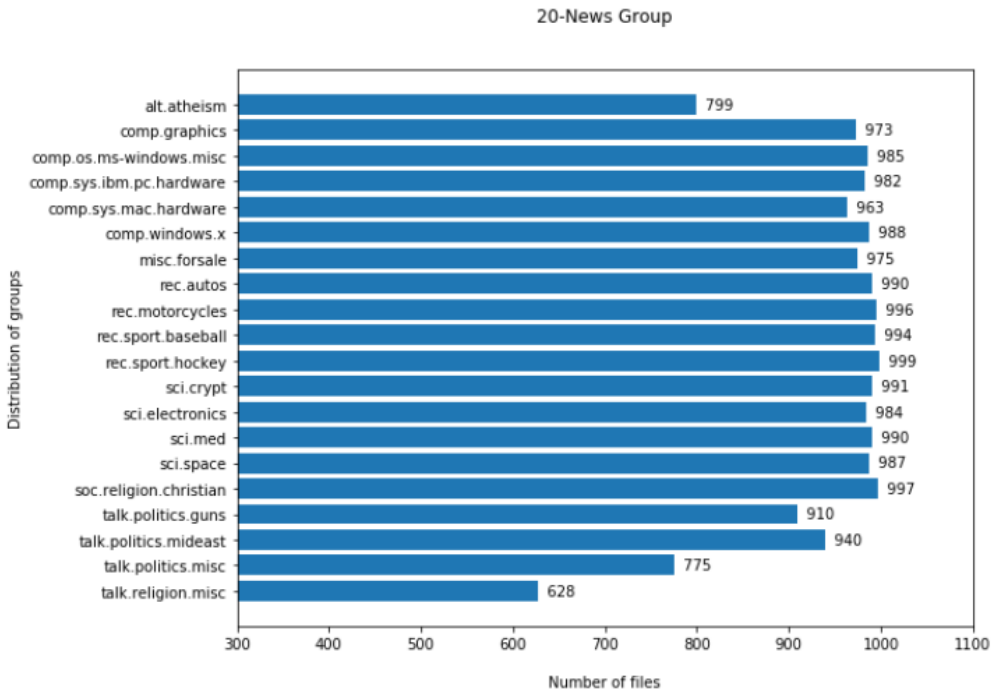


Figure 7: 20 News Group – Distribution of files

Here by using the 2nd variant (i.e. without headers, footers and quotes), we not only can reduce the number of terms, we also can reduce the overall processing time. And one important factor we found is that each and every file in this text corpus has headers, which points to the group type it belongs to. Headers are subject attached to every file; subject is always related to the document content, for example fever, breath, illness, ache are the words related to "sci.med" group. Therefore it helps the documents to form groups based on their original groups. But we achieved similar clusters even without including headers, footers and quotes. Example of headers present in an "alt.athesim" file is shown in the picture which is highlighted below, in the image from the word 'From' to the word "Version: 1.0" will not be included in the 2nd variant as these words are less important and also gives a hint that which group this document belongs:

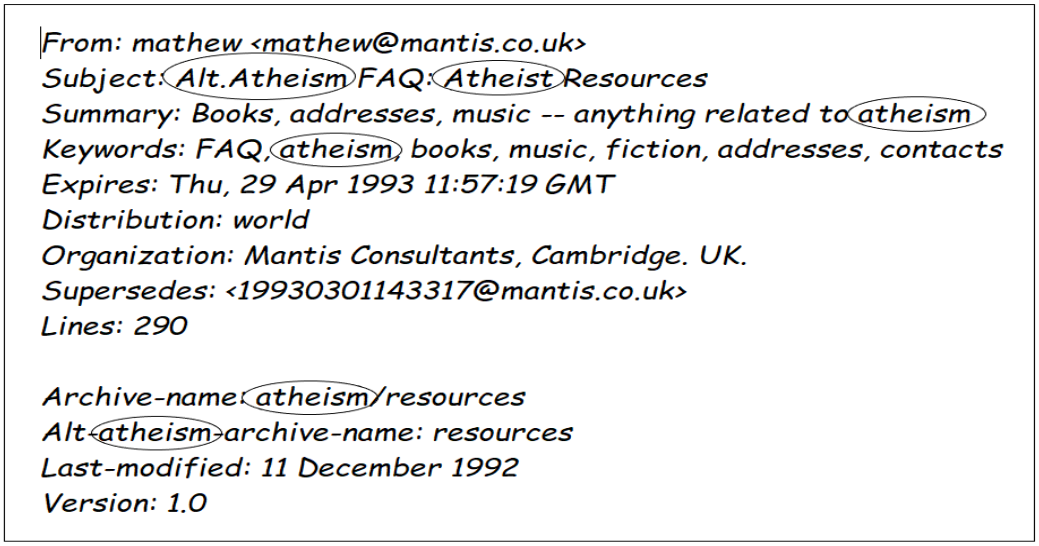


Figure 8: Example of a header – A file from "alt.atheism" group

Footers are details present in a file which is also connected to its group type. And quotes are some information related to the profession or a social message present below footers or one's email signatures. Example ".edu" can be an email signature which is used by a profession in the footer information. Sample file in the group "rec.motorcycles" is shown in the below Figure 7. Here we used both the variants and tried to get better clustering results even without any hints from these headers, footers and quotes.

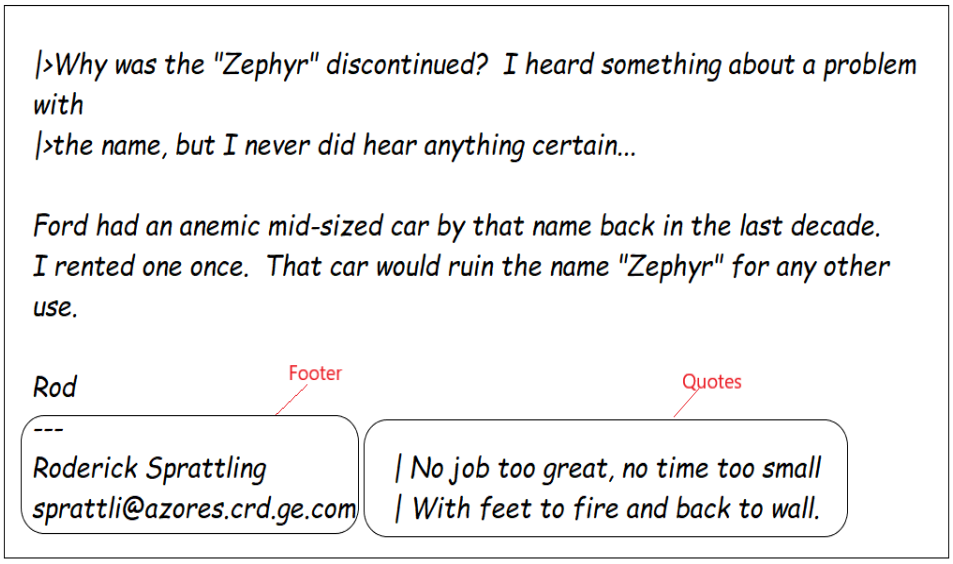


Figure 9: Example of a footer & quotes: A file from “rec.motorcycles” group

### 5.2.2 Reuters

Another famous dataset for text mining and analysis is the Reuters dataset. Reuters is a financial news wire service. We have selected the first version according to [17] in which it contains 90 categories (i.e. groups), but we have removed duplicate files from folders and preserved only the true files which results and corresponds to 80 groups of original classes which are based on topics.

| Category | Details |
|---|---|
| Name | Reuters |
| Original File size | 10.4 Mb |
| Data type | Text |
| Data source | unitn |
| Total number of files | 12897 |
| Avg. file count in a folder | 161.2 |
| No. of folders based on groups | 80 |

Table 8: Properties of Reuters dataset

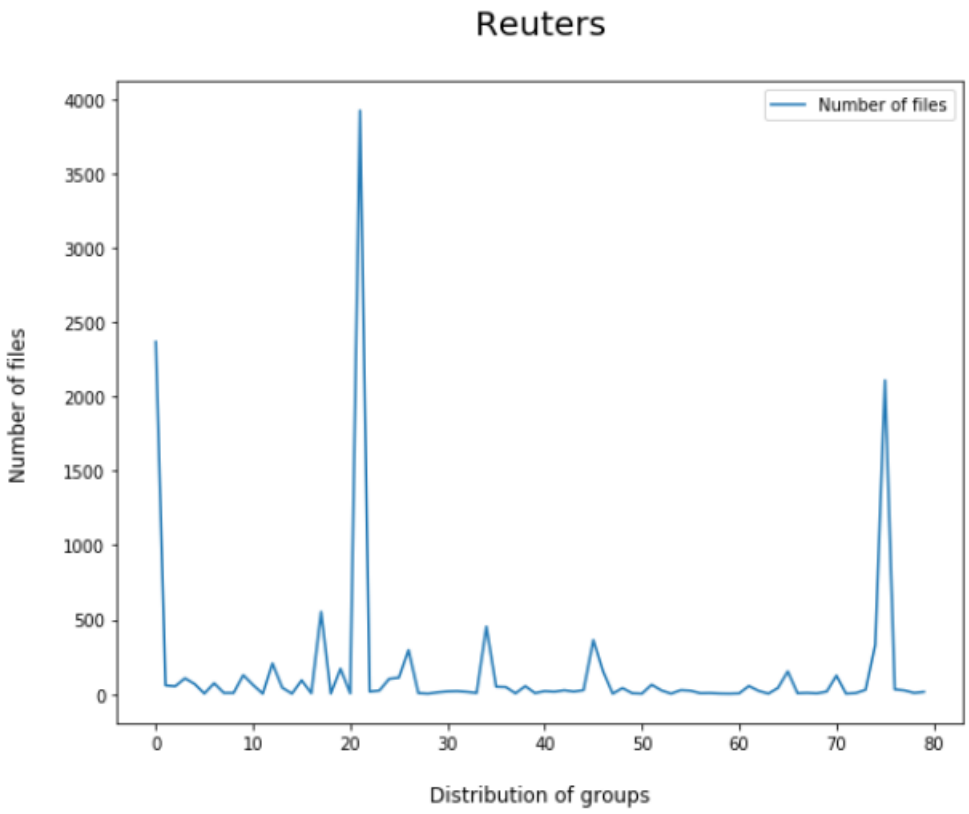


Figure 10: Reuters – Distribution of Files

## 5.3 Results

This section discusses the various experiments done with different settings which are split into four categories. The first two categories are done with k-means where sparse tfidf matrix is dimension reduced with the help of truncated-SVD into two different n numbers of terms. And the next two categories are for the Buckshot which is the combination of HAC and k-means with the same dimension reduction into two different n numbers of terms. Here we used the true k value for each dataset to calculate the entropy and purity scores to compare whether the dimension reduction provides any benefits to the two clustering algorithms. So for the 20 News group the k value is 20 and for the Reuters dataset k value is 80 respectively. The entropy score should usually lie between (0, $\log_2$ ($k_d$)), in which for 20 news group entropy score should be from 0 to $\log_2(20)$ i.e.:4.32 and for the reuters dataset the score should be from 0 to $\log_2(80)$ i.e. 6.32. and the purity scores should be between 0 to 1 where 1 is considered as perfect clustering and score close to 1 is considered as better clustering results.

The below table 9 shows the descriptive statistics of data pre-processing of 20 News group dataset.

| Variant no. | Type of Variant | Total no. of words | No. of Duplicate words | No. of Unique words (after pre-processing) |
|---|---|---|---|---|
| 1 | With headers, footers, quotes | 53,45,781 | 52,40,551 | 1,05,230 |
| 2 | Without headers, footers, quotes | 34,23,145 | 33,35,970 | 87,175 |

Table 9: 20 News group - Descriptive statistics of data pre-processing

The below table 10 shows the descriptive statistics of data pre-processing of Reuters dataset.

| Total no. of words | No. of Duplicate words | No. of Unique words (after pre-processing) |
|---|---|---|
| 17,28,786 | 16,96,683 | 32,103 |

Table 10: Reuters - Descriptive statistics of data pre-processing

### 5.3.1 Kmeans with 1000 terms

The sparse tfidf matrixes of 20 news group for two variants contains 1,05,230 and 87,176 terms and the Reuters dataset contains 32,103 terms are dimension reduced to 1000 terms in order to reduce the redundancy.

| External Validity Type | 20 – News Group | | Reuters |
|---|---|---|---|
| | With Headers, footers & quotes | Without Headers, footers & quotes | |
| Entropy | 0.5480 | 0.3939 | 0.5664 |
| Purity | 0.5866 | 0.4612 | 0.3350 |

Table 11: Kmeans with 1000 terms

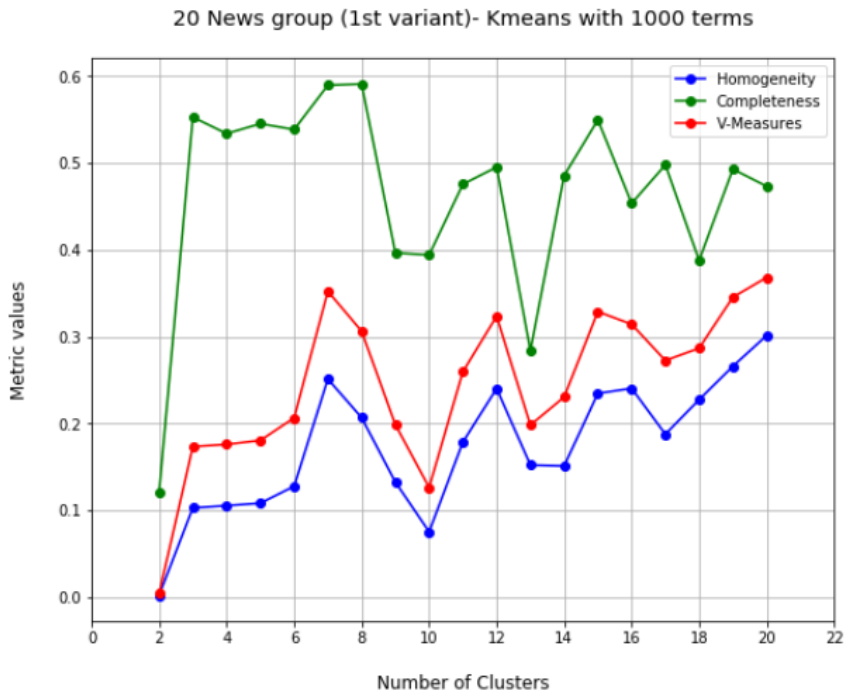


Figure 11: 20 News group (1st variant) - Kmeans with 1000 terms

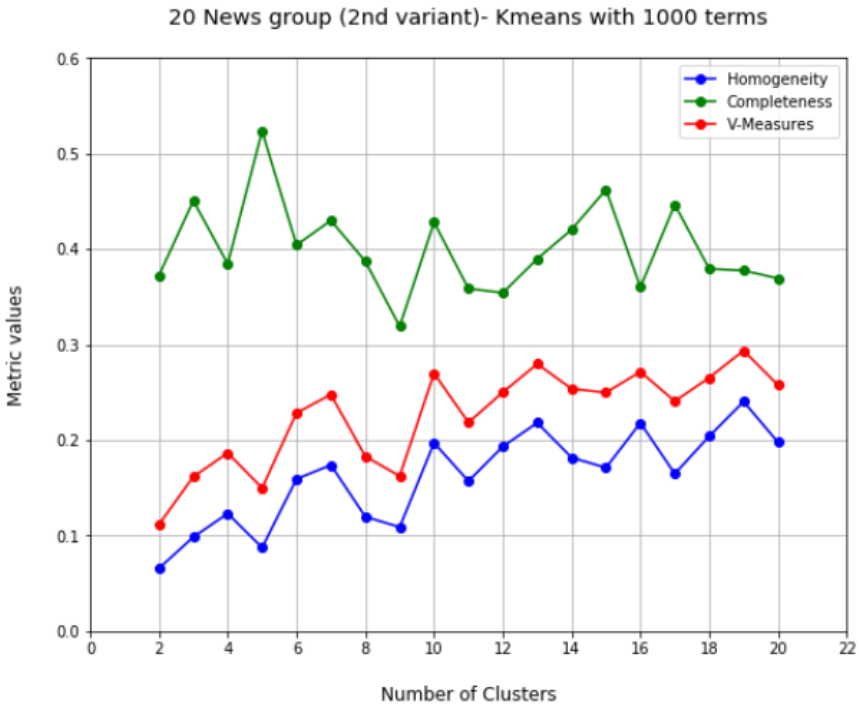


Figure 12: 20 News group (2nd variant) - Kmeans with 1000 terms

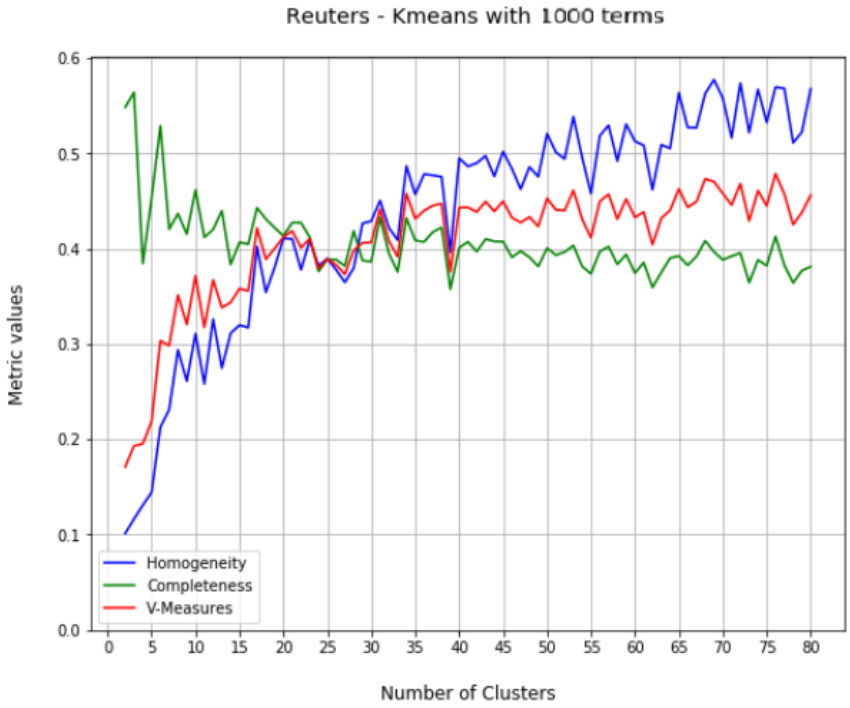


Figure 13: Reuters - Kmeans with 1000 terms

### 5.3.2 Kmeans with 100 terms

Here the sparse matrixes of both the datasets are dimension reduced to 100 terms in order to verify whether further dimension reduction produces better clustering results.

| External Validity Type | 20-News Group | | Reuters |
|---|---|---|---|
| | **With Headers, footers & quotes** | **Without Headers, footers & quotes** | |
| Entropy | 0.4293 | 0.2346 | 0.6379 |
| Purity | 0.4854 | 0.3689 | 0.2737 |

Table 12: Kmeans with 100 terms

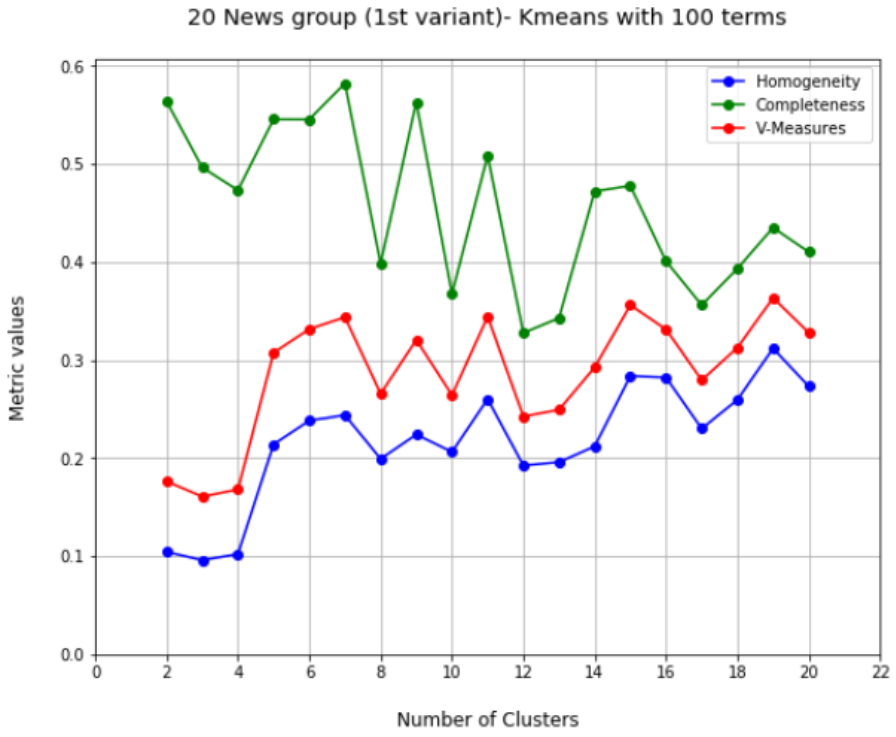


Figure 14: 20 News group (1st variant) - Kmeans with 100 terms

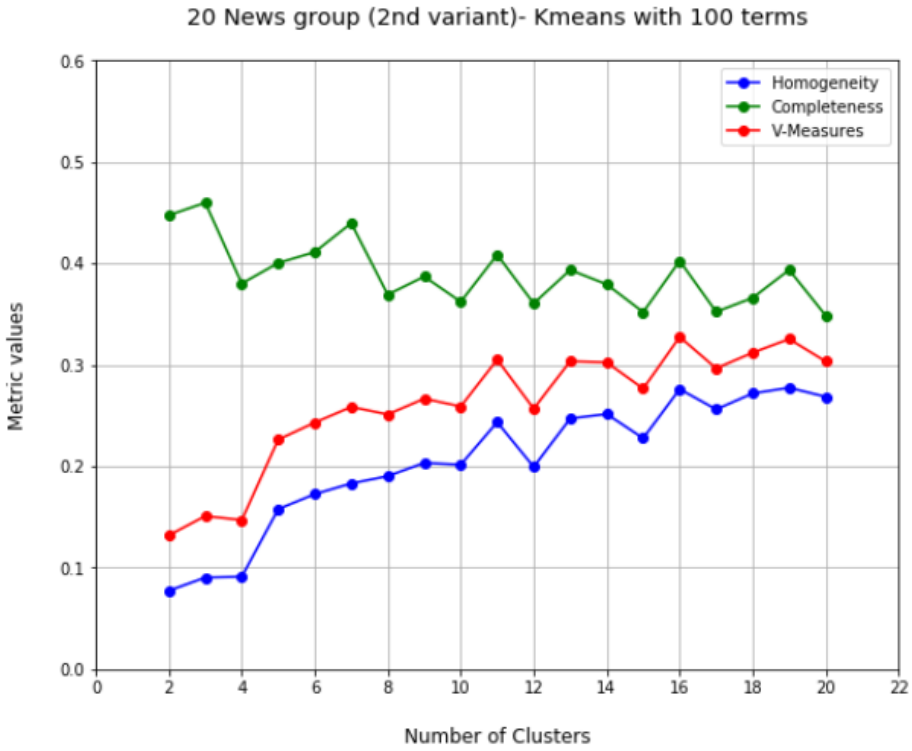


Figure 15: 20 News group (2nd variant) - Kmeans with 100 terms

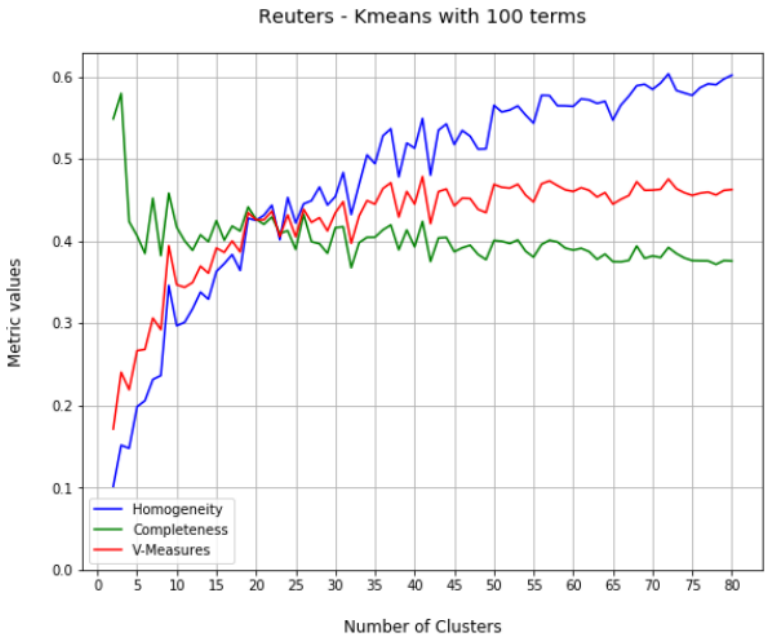


Figure 16: Reuters - Kmeans with 100 terms

### 5.3.3 Buckshot – 1000 terms

Here we have used sparse tfidf matrix and reduced its dimensions into 1000 terms for both the datasets. For the buckshot algorithm we have used $\sqrt{kn}$ for selecting the sample subset of documents $d$ for the hierarchical agglomerative clustering. So, for the 20 news group dataset which contains n number of documents (i.e.: 18846), we selected k = 20 which is the true k value of this dataset. Hence the sample subset of documents $d = \sqrt{kn} = \sqrt{20 * 18846} = 614$ and for the Reuters dataset which contains $n$ number of documents 12897 and true value $k$=80, thus $d = \sqrt{80 * 12897} = 1016$ respectively. Therefore for these $d$ numbers of randomly selected documents, the cosine distance measure is computed and their similarity matrix is used for the input for HAC.

| **External Validity Type** | **20-News Group** | | **Reuters** |
|---|---|---|---|
| | **With Headers, footers & quotes** | **Without Headers, footers & quotes** | |
| Entropy | 0.1957 | 0.3370 | 0.1119 |
| Purity | 0.6551 | 0.4795 | 0.3363 |

Table 13: Buckshot – 1000 terms

### 5.3.4 Buckshot – 100 terms

The main sparse tfidf matrix is dimension reduced to 100 terms for both these datasets in order to check whether more reduced terms produce better clustering results. Hence the *d* and *k* value remains the same, for 20 News group *k* and *d* are 20 and 614, for Reuters *k* and *d* are 80 and 1016 respectively.

| **External Validity Type** | **20-News Group** | | **Reuters** |
|---|---|---|---|
| | **With Headers, footers & quotes** | **Without Headers, footers & quotes** | |
| Entropy | 0.4395 | 0.3436 | 0.1440 |
| Purity | 0.5782 | 0.4073 | 0.2756 |

Table 14: Buckshot - 100 terms

## 5.4 Evaluation of Results

This section deals with various categories of experiments conducted for kmeans and Buckshot algorithms. Even though cluster validity score differs in each execution of both these algorithms we can see the results are at same phase for both these datasets and for all the categories. We will discuss the results obtained in detail from the above four categories.

### 5.4.1 Category 1

The results from kmeans with 1000 terms suggests that reducing the number of terms in both the datasets by approximately to a reasonable amount of terms gives better purity in cluster results. In 20 news group dataset even without using any hints from headers, footers and quotes for clustering, the variant 2 produces less entropy which is better compared to the variant 1. And we can also see the purity score slightly better compared to the variant 1 purity score.

### 5.4.2 Category 2

The results from kmeans with 100 terms suggest that reducing number of terms even more gives slightly lesser clustering validation scores. In 20-news group we see significant drop in both the scores in the variants and in Reuters, the entropy score is slightly higher and purity score drops lesser compared with 1000 terms, even though purity score should be higher and entropy score should be lower.

### 5.4.3 Category 3

The results from buckshot will be discussed in the categories 3 and 4. The buckshot with 1000 terms suggests there are better scores for 20 news group in entropy and slightly better scores even for Reuters where entropy is very close to 0 and purity is score is slightly higher for both these datasets when compared with kmeans with 1000 terms.

### 5.4.4 Category 4

The results for Buckshot with 100 terms produced slightly better scores compared with kmeans with 100 terms but when compared to buckshot with 1000 terms the scores are higher for entropy and we got lower purity. This suggests 1000 terms performed better not only for kmeans and even for buckshot algorithm and buckshot with 1000 terms produced better results from the overall categories.

# 6 CONCLUSION

This section discusses the most important results of work conducted. Also discusses the comparison of two main document clustering methods and how effectively these methods can be used with the dimension reduction techniques.

Text mining is has been around for years in various fields. Since, there is a high demand on the importance of document clustering. This work tries to analyse the different properties of the selected algorithms and also an effective use of the combinations of clustering methods with the use of dimension reduction techniques effectively.

For experimental process, we have selected 20 News group with two variants and Reuters in which both dataset contains thousands of text documents. Before working on the above mentioned datasets, the above mentioned algorithms were tested with small data in order to check the correctness of these mentioned algorithms. And then we have pre-processed the raw text documents from the dataset and only used lemmatization as one of the pre-processing procedure instead of stemming since lemmatization highly reduces the redundant text inside the corpus and finally we converted pre-processed data into numerically machine readable form of data using document term frequency matrix (TFIDF) and then used the matrix factorization techniques called Singular Value Decomposition (truncated-SVD) to reduce its noise and dimension of the sparse tfidf matrix, with two different n numbers of terms.

Also we performed various categories of k-Means clustering algorithm to detect clusters between similar text documents. We also performed Buckshot which is the combination of both clustering algorithm to compare different results and for both these algorithms we have used 100 numbers of iterations. We also obtained better clustering results for 20 news group in variant 2 even without using any information of which true groups these documents belongs. And we noticed that the buckshot with 1000 terms scored better overall. We also tried to reduce the number of terms into 2 in order to plot the cluster results visually using tsne from both these algorithms, but the cluster validity scores suffered heavily. Hence it is recommended that dimension reduction should be used to a reasonable amount in our case 1000 terms produced better results compared to 100 terms.

Here we noticed that the kmeans clustering worked better in terms of time and larger dimensions whereas buckshot algorithm performed even better compared to kmeans clustering. Finally we conclude that many algorithms are proposed for clustering similar text documents but still there is an open problem looking at the rate at which the demand of web applications and web documents are increasing every day, clustering these documents will always remain a high demand for these applications.

## 6.1 Future of Study

For future work, we can use graph based clustering method where nodes are considered as documents and edge weights are considered as similarity scores, since this clustering method requires similarity matrix which we achieved after pre-processing all the textual data from both the corpus used. One can use LSA method which uses Bag of Words (BoW) model, bag of words uses tfidf matrix, and in other words it uses another form of dimension reduction. Other clustering methods can be used to compare existing clustering methods and its results or using the above existing methods one can also extend it for systems like information retrieval. And one can also use the combination of information retrieval with other document clustering methods.

# Bibliography

[1] Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT, Press. Cambridge, MA: May 1999

[2] Dan Jurafsky, Speech and Language Processing, Pearson Education, 2014

[3] Bing Liu: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press; 1 edition (June 4, 2015)

[4] Charu C. Aggarwal, Machine learning for text. New York, NY: Springer Science Business Media, 2018. ISBN 978-3-319-73530-6.

[5] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. ACM SIGIR Conference, pp. 318 – 329, 1992.

[6] Larsen, B. and Aone, C., "Fast and Effective Text Mining Using Linear-time Document Clustering," KDD-99, SanDiego, California (1999).

[7] Steinbach, M., Karypis, G., Kumar, V., "A Comparison of Document Clustering Techniques"

[8] Mohammed J. Zaki, Wagner Meira Jr - Data Mining and Analysis: Fundamental Concepts and Algorithms, May 2014.

[9] Silke Wagner and Dorothea Wagner: Comparing Clusterings - An Overview, January 2007.

[10] Neepa Shah, Sunita Mahajan, PhD, Document Clustering: A Detailed Review, 2012.

[11] Huda Hamdan Ali, Lubna Emad Kadhum - K- Means Clustering Algorithm Applications in Data Mining and Pattern Recognition

[12] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu - Understanding of Internal Clustering Validation Measures, 2010.

[13] Comparison Of Purity And Entropy Of K-Means Clustering And Fuzzy C Means Clustering - Satya Chaitanya Sripada, Dr. M.Sreenivasa Rao, ISSN: 0976-5166

[14] Andrew Rosenberg and Julia Hirschberg, V-Measure: A conditional entropy-based external cluster evaluation measure, JAN 2007.

[15] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey- Scatter/Gather: a cluster-based approach to browsing large document collections.

[16] Document Clustering, Pankaj Jajoo, 2008.

[17] Reuters Dataset, University of Trento, http://disi.unitn.it/moschitti/corpora.htm

[18] Anaconda Edition for Python/R Data Science, https://www.anaconda.com/distribution/

[19] Evaluation of clustering, https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

[20] Overview of Text Similarity Metrics, https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50

[21] Clustering Performance Evaluation, https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

[22] Term frequency-inverse document frequency, http://www.tfidf.com/

[23] Official Page of t-SNE, https://lvdmaaten.github.io/tsne/

[24] Introduction to t-SNE, https://www.datacamp.com/community/tutorials/introduction-t-sne

[25] Introduction to Text mining, https://monkeylearn.com/what-is-text-mining/

[26] Introduction to Text Classification, https://monkeylearn.com/what-is-text-classification/